Probabilistic & Approximate Computing

Sasa Misailovic

UIUC

Reminder: Bayes' Rule

Belief Revision

Hypothesis

$$\Pr(\theta \mid x) = \frac{\Pr(x \mid \theta) \cdot \Pr(\theta)}{\Pr(x)}$$

Data

Bayes' Rule

Belief Revision

$$\mathrm{f}(\theta \mid x) = \frac{\mathrm{f}(x \mid \theta) \cdot \mathrm{f}(\theta)}{\mathrm{f}(x)}$$

Works for densities too!

Bayes' Rule

Belief Revision

$$\log \mathrm{f}(\theta \mid x) \sim \log \mathrm{f}(x \mid \theta) + \log \mathrm{f}(\theta)$$

Likelihood or Log-likelihood?

target = 0.0

```
X := Gaussian(0, 1);
```

target = $\mathcal{N}$_logpdf(X,0,1)

Likelihood or Log-likelihood?

target = 0.0

```
X := Gaussian(0, 1);
Y := Gaussian(X, 1);
```

target = $\mathcal{N}$_logpdf(X,0,1) + $\mathcal{N}$ _logpdf(Y,X,1)

Likelihood or Log-likelihood?

target = 0.0

```
X := Gaussian(0, 1);
Y := Gaussian(X, 1);
```

target = $\mathcal{N}$_logpdf(X,0,1) + $\mathcal{N}$ _logpdf(Y,X,1)

```
observe (Y, 0.5);
return X;
```

The probability that a continuous random variable has a particular value is equal to 0 (and logpdf(0) = -inf)

Likelihood or Log-likelihood?

target = 0.0

```
X := Gaussian(0, 1);
Y := Gaussian(X, 1);
```

target = $\mathcal{N}$_logpdf(X,0,1) + $\mathcal{N}$ _logpdf(Y,X,1)

```
observe (Y, 0.5);
```

```
factor (Gaussian.score(0.5, Y));
```

Likelihood or Log-likelihood?

target = 0.0

```
X := Gaussian(0, 1);
Y := Gaussian(X, 1);
```

target = $\mathcal{N}$_logpdf(X,0,1) + $\mathcal{N}$ _logpdf(Y,X,1)

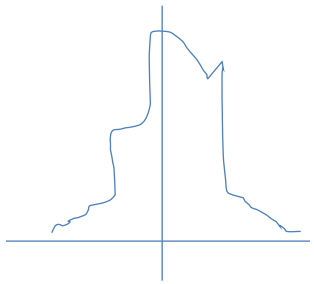
```
factor (Gaussian.score(0.5, Y));
```

target = $\mathcal{N}$_logpdf(X,0,1) + $\mathcal{N}$_logpdf(Y,X,1)
+ $\mathcal{N}$_logpdf(0.5,X,1)

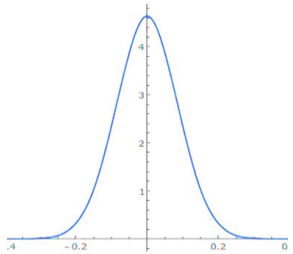
Approaches to Probabilistic Inference

Exact and Approximate

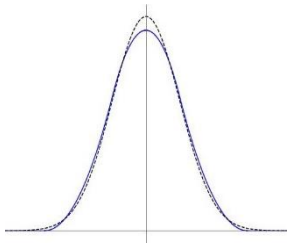
- **Sampling**
- (Rejection – Church)
- (MCMC – Church & Stan & R2)



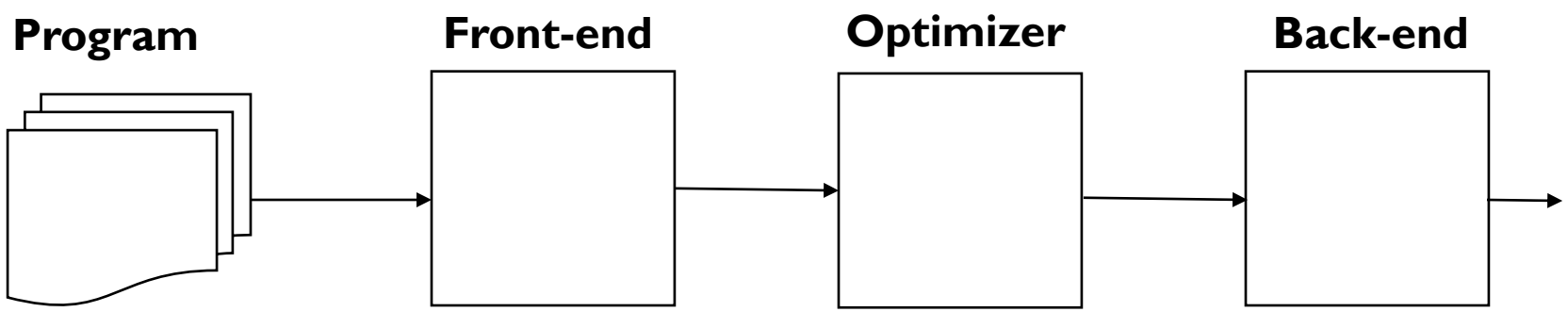
- **Variational Inference**
- (Fun & Infer.NET)



- **Exact Symbolic Inference**
- (PSI & Hakaru)

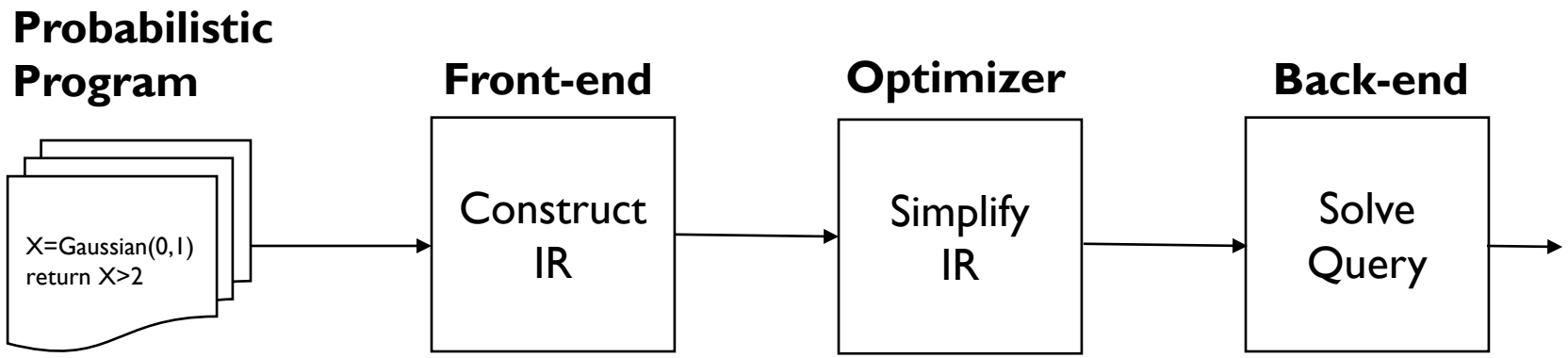


PSI Overview



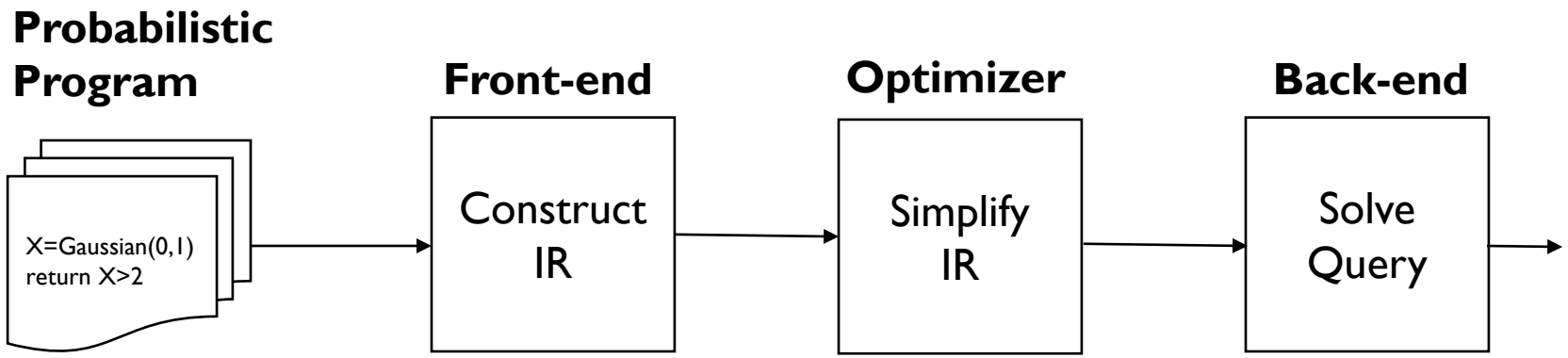
T. Gehr, S. Misailovic, M. Vechev; PSI: Exact Symbolic Inference for Probabilistic Programs (CAV 2016)

PSI Overview



$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2] \\
& \mid \int_{\mathbb{R}} \mathrm{d}x \, e[\![x]\!] \mid \sum_{x \in \mathbb{Z}} e[\![x]\!] \mid \varphi(e_1, \ldots, e_n)
\end{aligned}
$$

PSI Overview



- Basic algebraic simplifications

  $x + x \to 2 \cdot x, \quad x \cdot x \to x^2, \ldots$

- Simplifications on constraints

  $[x = 0] + [x \neq 0] \to 1, \quad [x \leq 0] \cdot [0 \leq x] \to [x = 0],$

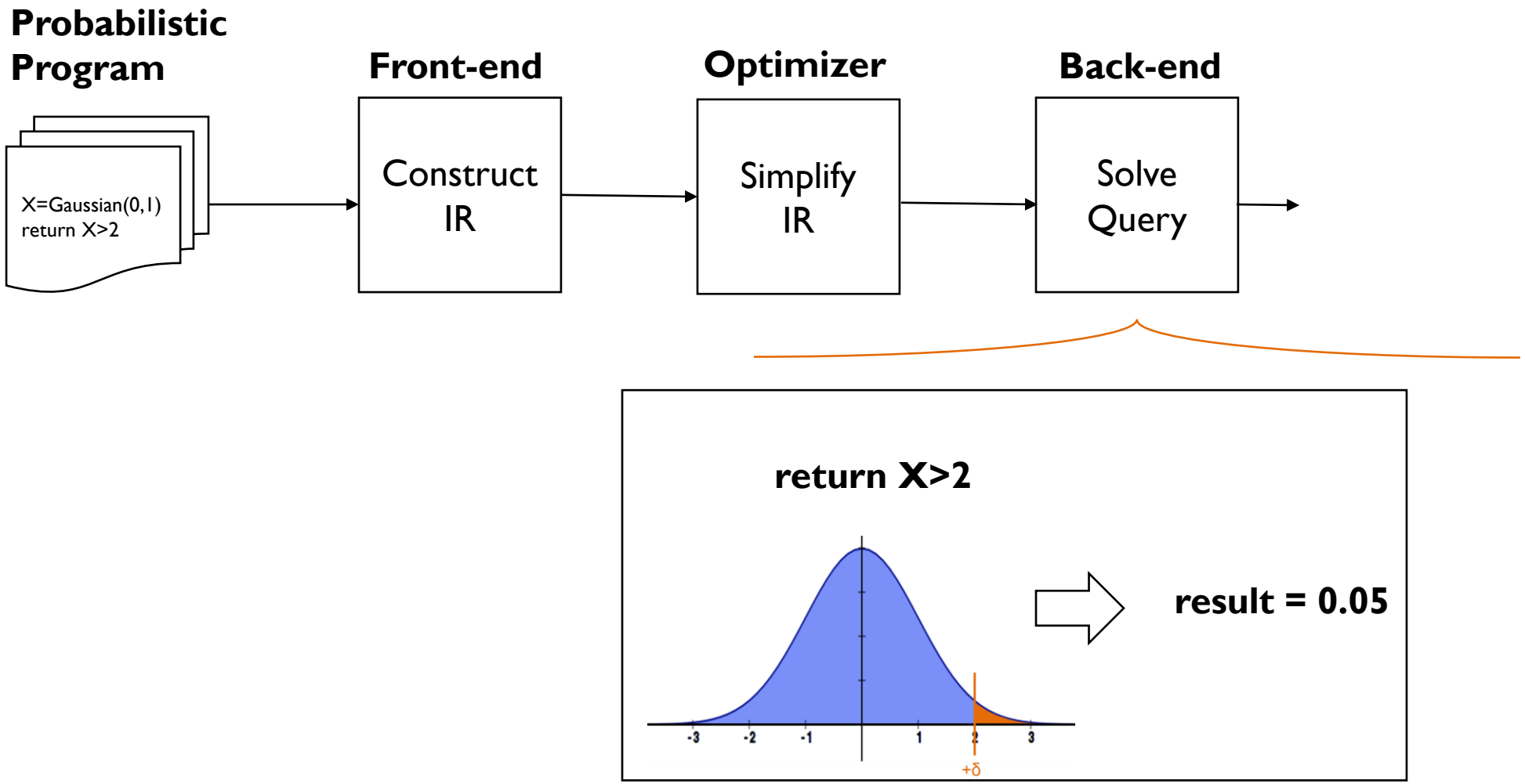
  $\delta(x) \cdot [1 \leq x] \to 0, \ldots$

- Symbolic Integration

  $\int_{\mathbb{R}} \mathrm{d}x \int_{\mathbb{R}} \mathrm{d}y\, [0 \leq x] \cdot [x \leq 1] \cdot [0 \leq y] \cdot [y \leq 1] \cdot \delta(z - x \cdot y) \to$

  $-[0 < z] \cdot [z \leq 1] \cdot \log(z), \ldots$

PSI Overview



PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2]
\end{aligned}
$$

Encodes probability density functions

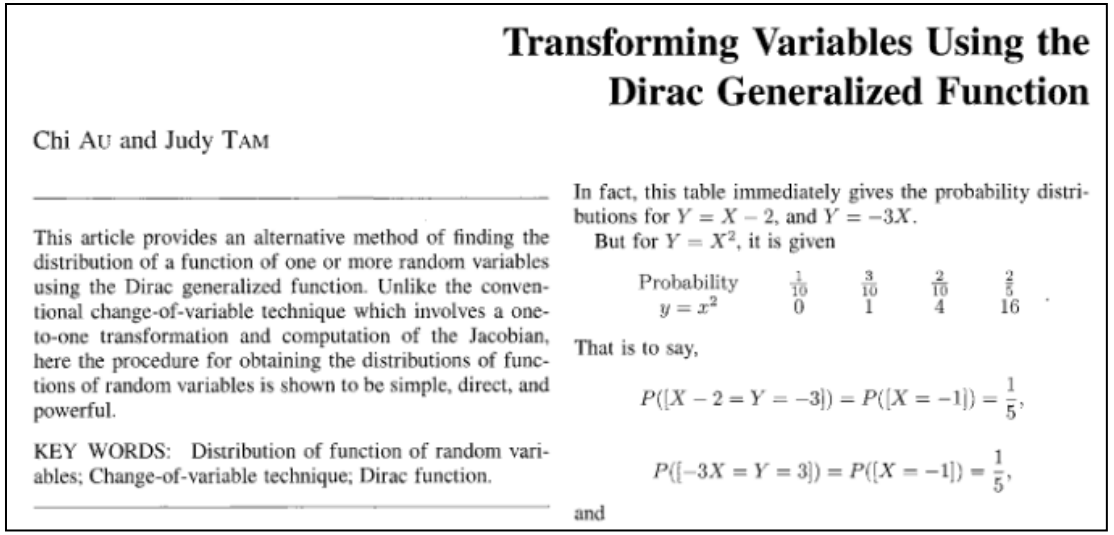
PSI

PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2]
\end{aligned}
$$

$$
\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x)\, dx = 1
$$

Encodes probability density functions



Transforming Variables Using the Dirac Generalized Function

Chi Au and Judy Tam

This article provides an alternative method of finding the distribution of a function of one or more random variables using the Dirac generalized function. Unlike the conventional change-of-variable technique which involves a one-to-one transformation and computation of the Jacobian, here the procedure for obtaining the distributions of functions of random variables is shown to be simple, direct, and powerful.

KEY WORDS: Distribution of function of random variables; Change-of-variable technique; Dirac function.

In fact, this table immediately gives the probability distributions for $Y = X - 2$, and $Y = -3X$.

But for $Y = X^2$, it is given

Probability	$\frac{1}{10}$	$\frac{3}{10}$	$\frac{2}{10}$	$\frac{2}{5}$
$y = x^2$	0	1	4	16

That is to say,

$$P([X - 2 = Y = -3]) = P([X = -1]) = \frac{1}{5},$$

$$P([-3X = Y = 3]) = P([X = -1]) = \frac{1}{5},$$

and

PSI

PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2]
\end{aligned}
$$

Encodes probability density functions

$$\mathsf{Bernoulli}(x; p) = p \cdot \delta(1 - x) + (1 - p) \cdot \delta(x)$$

PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2]
\end{aligned}
$$

Encodes probability density functions

$$\mathsf{Bernoulli}(x; p) = p \cdot \delta(1 - x) + (1 - p) \cdot \delta(x)$$

$$\mathsf{Gauss}(x; \mu, \nu) = [\nu = 0] \cdot \delta(x - \mu) + [\nu \neq 0] \cdot \frac{\mathsf{e}^{-(x-\mu)^2/(2\nu)}}{(2\pi\nu)^{\frac{1}{2}}}$$

PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2] \\
& \mid \int_{\mathbb{R}} \mathsf{d}x \, e[\![x]\!] \mid \sum_{x \in \mathbb{Z}} e[\![x]\!] \mid \varphi(e_1, \ldots, e_n)
\end{aligned}
$$

Encodes probability density functions

$$\mathsf{Bernoulli}(x; p) = p \cdot \delta(1 - x) + (1 - p) \cdot \delta(x)$$

$$\mathsf{Gauss}(x; \mu, \nu) = [\nu = 0] \cdot \delta(x - \mu) + [\nu \neq 0] \cdot \frac{\mathsf{e}^{-(x-\mu)^2/(2\nu)}}{(2\pi\nu)^{\frac{1}{2}}}$$

PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2] \\
& \mid \int_{\mathbb{R}} \mathsf{d}x \, e[\![x]\!] \mid \sum_{x \in \mathbb{Z}} e[\![x]\!] \mid \varphi(e_1, \ldots, e_n)
\end{aligned}
$$

Encodes probability density functions

$$\mathsf{Bernoulli}(x; p) = p \cdot \delta(1 - x) + (1 - p) \cdot \delta(x)$$

$$\mathsf{Gauss}(x; \mu, \nu) = [\nu = 0] \cdot \delta(x - \mu) + [\nu \neq 0] \cdot \frac{\mathsf{e}^{-(x-\mu)^2/(2\nu)}}{(2\pi\nu)^{\frac{1}{2}}}$$

$$\mathsf{UniformInt}(x; a, b) = \frac{\sum_{x' \in \mathbb{Z}} \delta(x - x') \cdot [a \leq x'] \cdot [x' \leq b]}{\sum_{x' \in \mathbb{Z}} [a \leq x'] \cdot [x' \leq b]}$$

PSI's Symbolic Domain

$$
\begin{aligned}
e \in E ::= \quad & x \mid \mathsf{e} \mid \pi \mid 0 \mid 1 \mid 2 \mid \ldots \\
& \mid \log(e) \mid -e \mid e_1 + \ldots + e_n \mid e_1 \cdot \ldots \cdot e_n \mid e_1^{e_2} \\
& \mid \delta(e) \mid [e_1 = e_2] \mid [e_1 \leq e_2] \mid [e_1 \neq e_2] \mid [e_1 < e_2] \\
& \mid \int_{\mathbb{R}} \mathsf{d}x \, e[\![x]\!] \mid \sum_{x \in \mathbb{Z}} e[\![x]\!] \mid \varphi(e_1, \ldots, e_n) \\
& \mid (\mathsf{d}/\mathsf{d}x)^{-1}[\mathsf{e}^{-x^2}](e) \quad \text{(Error function)}
\end{aligned}
$$

Encodes probability density functions

$$\mathsf{Bernoulli}(x; p) = p \cdot \delta(1 - x) + (1 - p) \cdot \delta(x)$$

$$\mathsf{Gauss}(x; \mu, \nu) = [\nu = 0] \cdot \delta(x - \mu) + [\nu \neq 0] \cdot \frac{\mathsf{e}^{-(x-\mu)^2/(2\nu)}}{(2\pi\nu)^{\frac{1}{2}}}$$

$$\mathsf{UniformInt}(x; a, b) = \frac{\sum_{x' \in \mathbb{Z}} \delta(x - x') \cdot [a \leq x'] \cdot [x' \leq b]}{\sum_{x' \in \mathbb{Z}} [a \leq x'] \cdot [x' \leq b]}$$

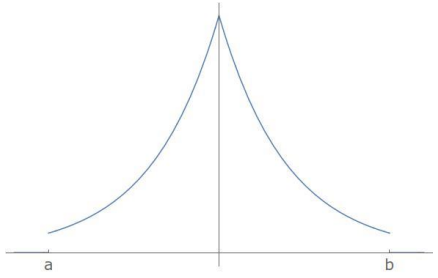
How About Sum of Three Variables?

$Z = X1 + X2 + X3$

X1, X2, X3
i.i.d.

Z

Truncated Laplace

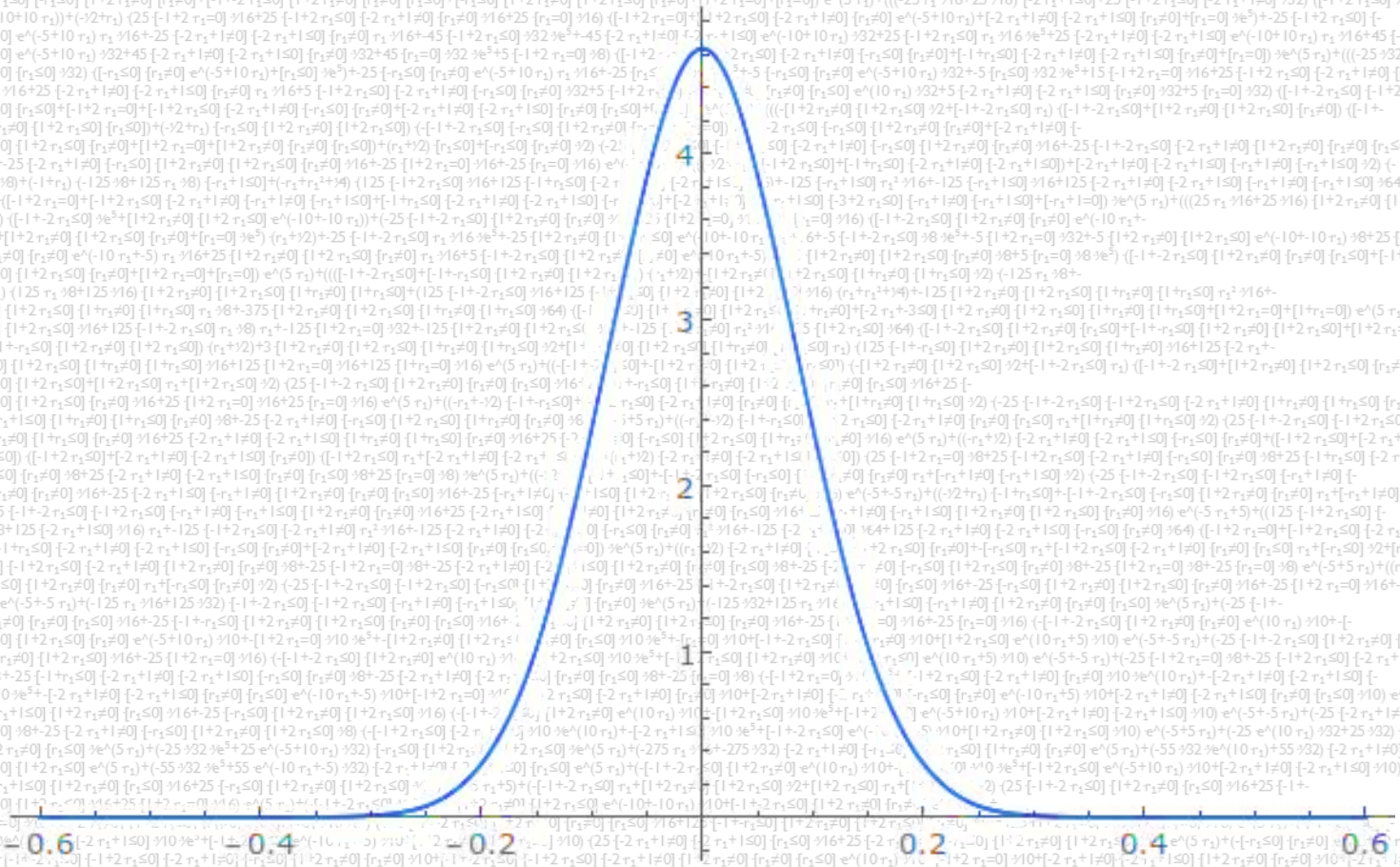


?

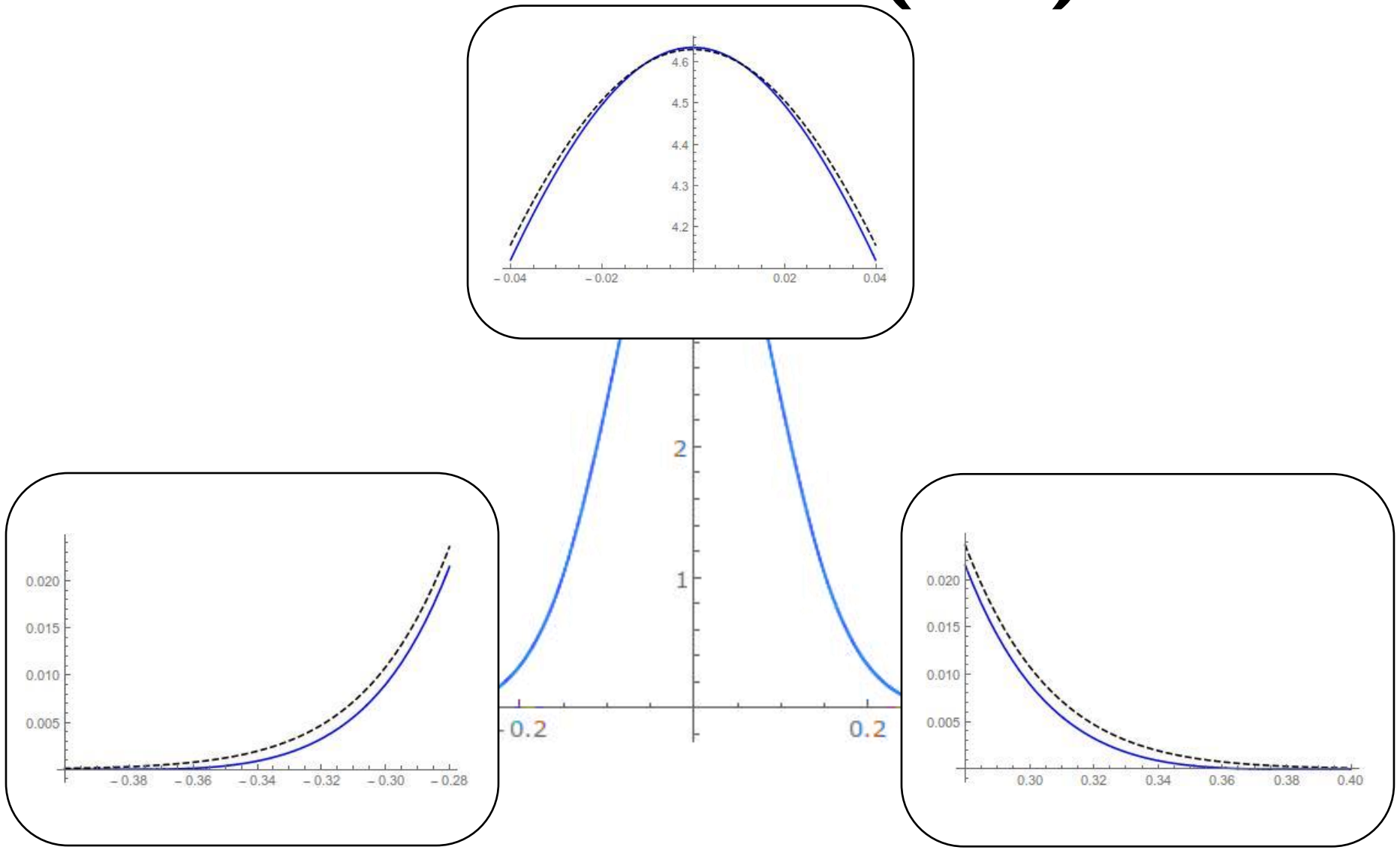
Exact Final Distribution (PSI)

p(r1) = ((((-25 r1 ⅟16+-25 ⅟32) ⋅[r1≤0]+-25 ⋅[-r1≤0] ⋅[r1≠0] ⅟32) ⋅([-r1≤0] ⅇ^5+[r1≠0] ⋅[r1≤0] ⅇ^(-10 r1+-5))+-25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] r1 ⅟16+-25 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] r1 ⅟16+-5 ⋅[-r1≤0] ⅟32 ⅇ^5+-5 ⋅[r1≠0] ⋅[r1≤0] ⅇ^(-10 r1+-5) ⅟32+15 ⋅[1+2 r1=0] ⅟16+25 ⋅[-r1≤0] r1 ⅟16 ⅇ^5+25 ⋅[r1≠0] ⋅[r1≤0] ⅇ^(-10 r1+-5) r1 ⅟16+5 ⋅[-1+-2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟32 ⅇ^(10 r1)+5 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟32+5 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟32+5 ⋅[r1=0] ⅟32) ⋅([-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0]+[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0]+[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0]+[1+2 r1=0]+[r1=0]) ⅇ^(5 r1)+(((-25 r1 ⅟16+25 ⅟16) ⋅[-2 r1+1≤0]+25 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⅟32) ⋅([-1+2 r1≤0] ⅇ^5+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⅇ^(-10+10 r1))+(-⅟2+r1) ⋅(25 ⋅[-1+2 r1=0] ⅟16+25 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅟16+25 ⋅[r1=0] ⅟16) ⋅([-1+2 r1=0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅇ^(-5+10 r1)+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0]+[r1=0] ⅇ^5)+-25 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅇ^(-5+10 r1) r1 ⅟16+-25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] r1 ⅟16+-45 ⋅[-1+2 r1≤0] ⅟32 ⅇ^5+-45 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⅇ^(-10+10 r1) ⅟32+25 ⋅[-1+2 r1≤0] r1 ⅟16 ⅇ^5+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⅇ^(-10+10 r1) r1 ⅟16+45 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅇ^(-5+10 r1) ⅟32+45 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] ⅟32+45 ⋅[r1=0] ⅟32 ⅇ^5+5 ⋅[-1+2 r1=0] ⅟8) ⋅([-1+2 r1=0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0]+[-1+r1≤0] ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0]+[r1=0]) ⅇ^(5 r1)+(((-25 ⅟32+25 r1 ⅟16) ⋅[-r1≤0]+-25 ⋅[r1≠0] ⋅[r1≤0] ⅟32) ⋅([-r1≤0] ⋅[r1≠0] ⅇ^(-5+10 r1)+[r1≤0] ⅇ^5)+-25 ⋅[-r1≤0] ⋅[r1≠0] ⅇ^(-5+10 r1) r1 ⅟16+-25 ⋅[r1≤0] ⋅[r1≠0] ⅇ^(-5+10 r1) ⅟32+-5 ⋅[r1≤0] ⅟32 ⅇ^5+15 ⋅[-1+2 r1=0] ⅟16+25 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0] r1 ⅟16+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] r1 ⅟16+5 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0] ⅟32+5 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅇ^(10 r1) ⅟32+5 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] ⅟32+5 ⋅[r1=0] ⅟32) ⋅([-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0]+[-1+2 r1=0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0]+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0]+[r1=0]) ⅇ^(5 r1)+(((-[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟2+[-1+-2 r1≤0] r1) ⋅([-1+-2 r1≤0]+[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0]) ⋅([-1+-2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0]+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0]) ⅟32+5 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅇ^(10 r1) ⅟32+5 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] ⅟32+5 ⋅[r1=0] ⅟32) ⋅([-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0]+[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0]+[r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0]+[1+2 r1=0]+[r1=0]) ⅇ^(5 r1)+(((-25 ⅟32+25 r1 ⅟16) ⋅[-r1≤0]+-25 ⋅[r1≠0] ⋅[r1≤0] ⅟32) ⋅([-r1≤0] ⋅[r1≠0] ⅇ^(-5+10 r1)+[r1≤0] ⅇ^5)+-25 ⋅[-r1≤0] ⋅[r1≠0] ⅇ^(-5+10 r1) r1 ⅟16+-25 ⋅[r1≤0] ⋅[r1≠0] ⅇ^(-5+10 r1) ⅟32+-5 ⋅[r1≤0] ⅟32 ⅇ^5+15 ⋅[-1+2 r1=0] ⅟16+25 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0] r1 ⅟16+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] r1 ⅟16+5 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0] ⅟32+5 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅇ^(10 r1) ⅟32+5 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] ⅟32+5 ⋅[r1=0] ⅟32) ⋅([-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0]+[-1+2 r1=0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0]+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0]+[r1=0]) ⅇ^(5 r1)+(((-[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟2+[-1+-2 r1≤0] r1) ⋅([-1+-2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0]+[-2 r1+1≠0] ⋅[-2 r1+1≤0]) ⋅(r1+⅟2)+-25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] r1 ⅟16+5 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅇ^(-10 r1+-5) ⅟8+5 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟8+5 ⋅[r1=0] ⅟8 ⅇ^5) ⋅([-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0]+[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0]+[1+2 r1=0]+[r1=0]) ⅇ^(5 r1)+(((-[1+-2 r1≤0]+[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0]) ⋅(r1+⅟2)+-[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⅟2) ⋅(-125 r1 ⅟8+-125 ⅟16)+(1+r1) ⋅(125 r1 ⅟8+125 ⅟16) ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0]+(125 ⋅[-1+-2 r1≤0] ⅟16+125 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟16) ⋅(r1+r1²+⅟4)+-125 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] r1² ⅟16+-125 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⅟64+-125 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] r1² ⅟16+125 ⋅[1+2 r1≤0] ⅟64) ⋅([-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0]+[1+2 r1=0]+[1+r1=0]) ⅇ^(5 r1)+(((-125 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟16+125 ⋅[-1+-2 r1≤0] r1 ⅟8) r1+-125 ⋅[1+2 r1=0] ⅟32+-125 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟64+-125 ⋅[1+2 r1≠0] r1² ⅟16+125 ⋅[1+2 r1≤0] ⅟64) ⋅([-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≤0]+[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0]+[1+2 r1=0]) ⅇ^(5 r1)+(((-[1+-2 r1≤0]+-[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0]) ⋅(r1+⅟2)+3 ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⅟2+[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] r1) ⋅(125 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⅟16+125 ⋅[-2 r1+3≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⅟16+125 ⋅[1+2 r1=0] ⅟16+125 ⋅[1+r1=0] ⅟16) ⅇ^(5 r1)+(((-[-1+-2 r1≤0]+-[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≤0]) ⋅(-[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟2+[-1+-2 r1≤0] r1) ⋅([-1+-2 r1≤0]+[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0])+(-r1+⅟2) ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0]+[1+2 r1≤0] r1+[1+2 r1≤0] ⅟2) ⋅(25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[1+2 r1=0] ⅟16+25 ⋅[r1=0] ⅟16) ⅇ^(5 r1)+((-r1+-⅟2) ⋅[-1+-r1≤0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0] r1+[1+r1≠0] ⋅[1+r1≤0] ⅟2) ⋅(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅟8+-25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅟8+-25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅟8+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0] ⅟8+25 ⋅[r1=0] ⅟8) ⅇ^(5 r1)+((-⅟2+r1) ⋅[-1+r1≤0]+[-1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] r1+[-r1+1≠0] ⋅[-r1+1≤0] ⅟2) ⋅(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-r1+1≠0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[-r1+1≠0] ⋅[-r1+1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16) ⅇ^(-5+-5 r1)+((-⅟2+r1) ⋅[-1+r1≤0]+[-1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] r1+[-r1+1≠0] ⋅[-r1+1≤0] ⅟2) ⋅(25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-r1+1≠0] ⋅[-r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟16+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[-r1+1≠0] ⋅[-r1+1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16) ⅇ^(-5 r1+5)+((⅟2+r1) ⋅[-r1+1≤0]+[-1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] r1+[-r1+1≠0] ⋅[-r1+1≤0] ⅟2) ⋅(25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-r1+1≠0] ⋅[-r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟16+25 ⋅[-2 r1+1≤0] ⋅[-r1+1≠0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟16+25 ⋅[-r1+1≠0] ⋅[-r1+1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16) ⅇ^(-5 r1+5)+((⅟125 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] r1 ⅟8+125 ⋅[-2 r1+1≤0] ⅟16) r1+-125 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] r1² ⅟16+-125 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[-r1≤0] ⋅[r1≠0] r1² ⅟16+-125 ⋅[-2 r1+1≤0] ⅟64+125 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[-r1≤0] ⋅[r1≠0] ⅟64) ⋅([-1+2 r1=0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[r1≠0]+[-1+r1≤0] ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≠0]+[r1=0]) ⅇ^(5 r1)+((r1+⅟2) ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0]+[-r1≤0] r1+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⋅[r1≤0] r1+[-r1≤0] ⅟2+[r1≠0] ⋅[r1≤0] ⅟2) ⋅(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟8+-25 ⋅[-1+2 r1=0] ⅟8+-25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟8+-25 ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟8+-25 ⋅[1+2 r1=0] ⅟8+-25 ⋅[r1=0] ⅟8) ⅇ^(-5+5 r1)+((r1+⅟2) ⋅[-r1≤0]+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] r1+[-r1≤0] ⅟2+[r1≠0] ⋅[r1≤0] ⅟2) ⋅(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟8+-25 ⋅[-1+2 r1=0] ⅟8+-25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟8+-25 ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟8+-25 ⋅[1+2 r1=0] ⅟8+-25 ⋅[r1=0] ⅟8) ⅇ^(-5+5 r1)+((r1+⅟2) ⋅[r1≤0]+-[-1+-2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] r1+[-r1≤0] ⅟2+[r1≠0] ⋅[r1≤0] ⅟2) ⋅(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟16+-25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+-25 ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+-25 ⋅[1+2 r1=0] ⅟16+-25 ⋅[r1=0] ⅟16) ⅇ^(-5 r1+5)+((-[1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟2+[1+2 r1≤0] r1+[1+2 r1≤0] ⅟2) ⋅(25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[1+2 r1=0] ⅟16+25 ⋅[r1=0] ⅟16) ⅇ^(5 r1)+((-[-1+-2 r1≤0] ⅟10 ⅇ^5+-[1+2 r1≠0] ⋅[1+2 r1≤0] ⅇ^(-10+-10 r1) ⅟10+[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅇ^(-10 r1+-5) ⅟10+[1+2 r1=0] ⅟10+[1+2 r1≤0] ⅟10 ⅇ^5+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟10 ⅇ^(10 r1)+[-1+2 r1≤0] ⅟10) ⅇ^(-5+5 r1)+(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟8+25 ⋅[-1+2 r1=0] ⅟8+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟8+25 ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟8+25 ⋅[1+2 r1=0] ⅟8+25 ⋅[r1=0] ⅟8) ⅇ^(5 r1)+(-[-r1≤0] ⅟10 ⅇ^5+-[r1≠0] ⋅[r1≤0] ⅇ^(-10 r1+-5) ⅟10+[-1+-2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅟10 ⅇ^(10 r1)+[-1+-2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟16+25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟10 ⅇ^(10 r1)+[-1+2 r1≤0] ⅟10) ⅇ^(-5+5 r1)+(-25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟16+25 ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[1+2 r1=0] ⅟16+25 ⋅[r1=0] ⅟16) ⅇ^(5 r1)+(-[-r1≤0] ⅟10 ⅇ^5+-[r1≠0] ⋅[r1≤0] ⅇ^(-10 r1+-5) ⅟10+[-1+-2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅟10 ⅇ^(10 r1)+[-1+2 r1≤0] ⅟10) ⅇ^(-5+5 r1)+(-25 ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟8+25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⅟8+25 ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟8+25 ⋅[1+2 r1=0] ⅟8+25 ⋅[r1=0] ⅟8) ⅇ^(5 r1)+(-[-1+2 r1=0] ⅟10 ⅇ^5+[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[r1≠0] ⅟10 ⅇ^(10 r1)+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⅟10) ⅇ^(-5+5 r1)+(-25 ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⅟32+25 ⅟32) ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≤0] ⅇ^(5 r1)+(-25 ⅟32 ⅇ^5+25 ⅇ^(-5+10 r1) ⅟32) ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅇ^(5 r1)+(-275 r1 ⅟16+-275 ⅟32) ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≤0] ⋅[1+r1≠0] ⋅[r1≠0] ⅇ^(5 r1)+(-55 ⅟32 ⅇ^(10 r1)+55 ⅟32) ⋅[-2 r1+1≠0] ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅇ^(5 r1)+(-55 ⅟32 ⅇ^5+55 ⅇ^(-10 r1+-5) ⅟32) ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[r1≤0] ⅇ^(5 r1)+(-[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⅇ^(10 r1) ⅟10+[-1+2 r1≤0] ⅟10 ⅇ^5+[-1+2 r1≤0] ⅇ^(-5+10 r1) ⅟10+[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⅟10) ⋅(25 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+2 r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟16) ⅇ^(-5 r1+5)+(-[-1+-2 r1≤0] r1+[1+2 r1≠0] ⋅[1+2 r1≤0] ⅟2+[1+2 r1≤0] r1+[1+2 r1≤0] ⅟2) ⋅(25 ⋅[-1+-2 r1≤0] ⋅[1+2 r1≠0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⋅[r1≤0] ⅟16+25 ⋅[-1+-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[-r1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟16+25 ⋅[1+2 r1=0] ⅟16+25 ⋅[r1=0] ⅟16) ⅇ^(5 r1)+(275 r1 ⅟16+275 ⅟32) ⋅[-1+-2 r1≤0] ⋅[-1+2 r1≤0] ⋅[-2 r1+1≠0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅇ^(5 r1)+125 ⋅[-r1+1≠0] ⋅[-r1+1≤0] ⋅[1+2 r1≠0] ⋅[1+2 r1≤0] ⋅[r1≠0] ⅟32 ⅇ^(5 r1)+275 ⋅[-2 r1+1≠0] ⋅[-2 r1+1≤0] ⋅[1+r1≠0] ⋅[1+r1≤0] ⋅[r1≠0] ⅇ^(5 r1) ⅟32) ⋅[-3 ⅇ^(5 ⅟2)+-ⅇ^(15 ⅟2)+1+3 ⅇ^5)

Exact Final Distribution (PSI)



Exact Final Distribution (PSI)



Continuous Example

- `X := Laplace(0,1)`
- `Y := Laplace(0,1)`
- `Z = X + Y`
- `observe Z > 2`
- `return X`

Continuous Example

- `X := Laplace(0,1)`
- `Y := Laplace(0,1)`
- `Z = X + Y`
- `observe Z > 2`
- `return X`

Variable Definition Rule:

0: Add variable to the state:

$$p() \rightarrow p(X)$$

1: Get the Laplace distribution expression

$$f(t_{tmp}) = \frac{\lambda}{2} e^{-\lambda|t_{tmp}-\mu|}$$

2: Relate it to the program's variable X

$$p(X) = \text{replace } t_{tmp} \text{ with } X \text{ in } f(t_{tmp})$$

Continuous Example

- `X := Laplace(0,1)`
- `Y := Laplace(0,1)`
- `Z = X + Y`
- `observe Z > 2`
- `return X`

Variable Definition Rule:

0: Add variable to the state:

$$p() \rightarrow p(X)$$

1: Get the Laplace distribution expression

$$f(t_{tmp}) = \frac{\lambda}{2} e^{-\lambda|t_{tmp}-\mu|}$$

2: Relate it to the program's variable X

$$p(X) = \int dt_{tmp} \cdot \delta[X - t_{tmp}] \cdot f(t_{tmp})$$

Continuous Example

- `X := Laplace(0,1)`

$$p(X) = \int dt_{tmp} \cdot \delta[X - t_{tmp}] \cdot \frac{\lambda}{2} e^{-\lambda|t_{tmp}-\mu|}$$

- `Y := Laplace(0,1)`

- `Z = X + Y`

- `observe Z > 2`

- `return X`

Integration with Dirac Deltas:

$$\int dt_{tmp} \cdot \delta[X - t_{tmp}] \cdot f(t_{tmp}) = f(X)$$

$$=$$

Simplification Rules

- **Basic Algebraic Simplifications**

  $x + x \to 2 \cdot x, \quad x \cdot x \to x^2, \ldots$

- **Simplifications of Constraints**

  $[x = 0] + [x \neq 0] \to 1, \quad [x \leq 0] \cdot [0 \leq x] \to [x = 0],$

  $\delta(x) \cdot [1 \leq x] \to 0, \ldots$

- **Guard Linearization**

  $\delta(y - x^2) \to [-y \leq 0] \cdot ([x = 0] \cdot \delta(y) + [x \neq 0] \cdot \frac{1}{2\sqrt{y}}(\delta(x - \sqrt{y}) + \delta(x + \sqrt{y})))$

- **Symbolic Integration**

  $\int_{\mathbb{R}} \mathrm{d}x \int_{\mathbb{R}} \mathrm{d}y \, [0 \leq x] \cdot [x \leq 1] \cdot [0 \leq y] \cdot [y \leq 1] \cdot \delta(z - x \cdot y) \to$

  $-[0 < z] \cdot [z \leq 1] \cdot \log(z), \ldots$

Continuous Example

- `X := Laplace(0,1)`

  $p(X) = \frac{\lambda}{2} e^{-\lambda|X-\mu|}$

- `Y := Laplace(0,1)`

- `Z = X + Y`

- `observe Z > 2`

- `return X`

Integration with Dirac Deltas:

$$\int dt_{tmp} \cdot \delta[X - t_{tmp}] \cdot f(t_{tmp}) = f(X)$$

$$=$$

Continuous Example

- `X := Laplace(0,1)`

  $p(X) = \frac{1}{2} e^{-|X|}$

- `Y := Laplace(0,1)`

- `Z = X + Y`

- `observe Z > 2`

- `return X`

Integration with Dirac Deltas:

$$\int dt_{tmp} \cdot \delta[X - t_{tmp}] \cdot f(t_{tmp}) = f(X)$$

$$=$$

Continuous Example

- `X := Laplace(0,1)`

  $p(X) = \frac{1}{2} e^{-|X|}$

- `Y := Laplace(0,1)`

  $p(X, Y) = \frac{1}{2} e^{-|X|} \cdot \frac{1}{2} e^{-|Y|}$

- `Z = X + Y`

- `observe Z > 2`

- `return X`

Continuous Example

- `X := Laplace(0,1)`

  $$p(X) = \frac{1}{2}e^{-|X|}$$

- `Y := Laplace(0,1)`

  $$p(X,Y) = \frac{1}{2}e^{-|X|} \cdot \frac{1}{2}e^{-|Y|}$$

- `Z = X + Y`

  $$p(X,Y,Z) = \frac{1}{2}e^{-|X|} \cdot \frac{1}{2}e^{-|Y|} \cdot \delta(Z - (X+Y))$$

- `observe Z > 2`

- `return X`

Assignment Rule (Deterministic):

0: Add variable Z to the state:

$$p(X,Y) \rightarrow p(X,Y,Z)$$

1: Relate the program's variables

$$p(X,Y,Z) = p(X,Y) \cdot \delta[Z - Expr(X,Y)]$$

Continuous Example

- `X := Laplace(0,1)`

  $$p(X) = \frac{1}{2}e^{-|X|}$$

- `Y := Laplace(0,1)`

  $$p(X,Y) = \frac{1}{2}e^{-|X|} \cdot \frac{1}{2}e^{-|Y|}$$

- `Z = X + Y`

  $$p(X,Y,Z) = \frac{1}{2}e^{-|X|} \cdot \frac{1}{2}e^{-|Y|} \cdot \delta(Z - (X+Y))$$

- `observe Z > 2`

  $$p(X,Y,Z|Z>2) \sim \frac{1}{2}e^{-|X|} \cdot \frac{1}{2}e^{-|Y|} \cdot \delta(Z - (X+Y)) \cdot [Z > 2]$$

- `return X`

Continuous Example

- `X := Laplace(0,1)`

  $p(X) = \frac{1}{2} e^{-|X|}$

- `Y := Laplace(0,1)`

  $p(X,Y) = \frac{1}{2} e^{-|X|} \cdot \frac{1}{2} e^{-|Y|}$

- `Z = X + Y`

  $p(X,Y,Z) = \frac{1}{2} e^{-|X|} \cdot \frac{1}{2} e^{-|Y|} \cdot \delta(Z - (X+Y))$

- `observe Z > 2`

  $p(X,Y,Z|Z>2) \sim \frac{1}{2} e^{-|X|} \cdot \frac{1}{2} e^{-|Y|} \cdot \delta(Z - (X+Y)) \cdot [Z > 2]$

- `return X`

  1. Marginalization
  2. Normalization

$p(X \mid Z > 2) \sim \int dY \int dZ \;\; p(X,Y,Z|Z>2)$

$Const = \int dX \int dY \int dZ \;\; p(X,Y,Z|Z>2)$

Continuous Example

1. Marginalization

$$p(X \mid Z > 2) \sim \int dY \int dZ \;\; p(X, Y, Z | Z > 2)$$

$$\sim \int dY \int dZ \;\; \frac{1}{2} e^{-|X|} \cdot \frac{1}{2} e^{-|Y|} \cdot \delta(Z - (X + Y)) \cdot [Z > 2]$$

$$\sim \int dY \;\; \frac{1}{2} e^{-|X|} \cdot \frac{1}{2} e^{-|Y|} \cdot [X + Y > 2]$$

$$\sim \frac{1}{4} e^{-|X|} \cdot \left(e^{X-2} \cdot [X \leq 2] + \left(2 - e^{2-X}\right) \cdot [X > 2] \right)$$

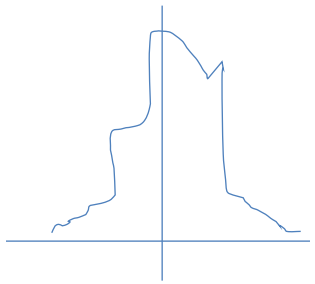
2. Normalization

$$Const = \int dx \int dy \int dz \;\; p(X, Y, Z | Z > 2) = e^{-2}$$

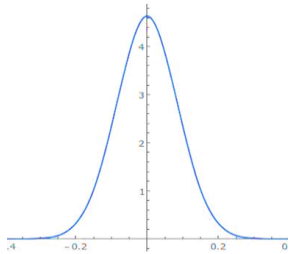
Approaches to Probabilistic Inference

Exact and Approximate

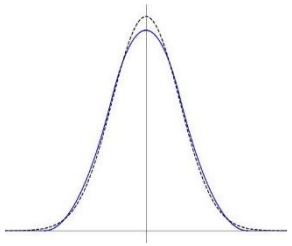
- **Sampling**
- (Rejection – Church)
- (MCMC – Church & Stan & R2)



- **Variational Inference**
- (Fun & Infer.NET)



- **Exact Symbolic Inference**
- (PSI & Hakaru)



Operational Semantics *(deterministic)*

- Simulates how the program executes on an abstract
- state-machine

X	Y
3	0

PC
32

```
31:   …
32:   y = x + 2
33:   …
```

X	Y
3	5

PC
33

- **Two flavors:**
- Small-step operational semantics
- *read from location pointed to by x, do addition, store to location pointed to by y*
- Big-step operational semantics
- *x + 2 evaluates to the value 5; store this value to the location pointed to by y*

Operational Semantics *(deterministic)*

- **Execution Stages:**
- Map the program and inputs to the ***initial configuration***

```
init:     (x, 3)
01:  def program(x){…}
```

X	Y
3	0

PC
01

- ***Execute the steps*** that represent the instructions of the abstract machine

```
32:  y = x + 2
```

X	Y
3	5

PC
33

- Map the ***final configuration*** (if it exists) to the output

```
33:  return y
```

X	Y
3	5

PC
33

Simple Deterministic Language

$$
\begin{array}{lcl}
x & \in & \text{Vars} \\
\mathcal{T} & ::= & \texttt{bool} \\
\textbf{uop} & ::= & \texttt{not} \\
\textbf{bop} & ::= & \texttt{and} \mid \texttt{or} \\
\mathcal{D} & ::= & \mid \mathcal{T}\, x_1, x_2, \ldots, x_n \\
\mathcal{E} & ::= & \\
 & & x \\
 & & \mid c \\
 & & \mid \mathcal{E}_1 \ \textbf{bop}\ \mathcal{E}_2 \\
 & & \mid \textbf{uop}\ \mathcal{E}_1
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{S} & ::= & \\
 & & x := \mathcal{E} \\
 & & \mid \mathcal{S}_1 ; \mathcal{S}_2 \\
 & & \mid \texttt{if}\ \mathcal{E}\ \texttt{then}\ \mathcal{S}_1\ \texttt{else}\ \mathcal{S}_2 \\
 & & \mid \texttt{while}\ \mathcal{E}\ \texttt{do}\ \mathcal{S}_1 \\
\mathcal{P} & ::= & \mathcal{D}\ \mathcal{S}
\end{array}
$$

Subset of Language from Claret et al. Bayesian inference using data flow analysis (FSE'13)

Operational Semantics

- Tuple: $S = (\mathcal{C}, \rightarrow, \mathcal{C}_{final}, \mathcal{I}, \mathcal{O})$
- $\mathcal{C}$: Set of configurations (e.g.,$\mathcal{C} = Stmt \times Stack \times Mem$)
- $\rightarrow$: Transition relation, which defines possible transitions between the configurations
  - Deterministic if for each start configuration $c \in \mathcal{C}$ there exists a single result configuration $c' \in \mathcal{C}$
  - Nondeterministic if there can be multiple $c \in \mathcal{C}$
- $\mathcal{C}_{final}$: Set of final configurations ($\mathcal{C}_{final} \subseteq \mathcal{C}$) in which the program successfully ends
- $\mathcal{I}$ maps program source and input to initial configuration $\mathcal{C}_0$
- $\mathcal{O}$ maps final configuration $\mathcal{C}_{final}$ to the output

Operational Semantics *(small step)*

- **Configuration:** $c \in \mathcal{C} ::= Stmt \times \Sigma$
- $\sigma \in \Sigma ::= (x_1 \rightarrow v_1, \ldots x_n \rightarrow v_n)$
- $Stmt$ – next statement or no statement to run $(\cdot)$
  x_i – variables, v_i – values

- **Final Configuration:**
- Has the form `(skip,` σ`) = (skip,` $(x_1 \rightarrow v_1, \ldots x_n \rightarrow v_n)$`)`
- Specifies configurations in which program terminated normally
- But, execution can get stuck (there is a statement for which there is no transition)
- Or, execution may never terminate (the execution loops infinitely)

Operational Semantics *(small step)*

For expressions: $\rightarrow_b \in Expr \times \Sigma \rightarrow Expr$

e1, e2 – expressions
v1, v2 – values

$(\texttt{x},\ \sigma) \rightarrow_b \sigma(x)$

Reading a variable

$(\texttt{uop v1},\ \sigma) \rightarrow_b \texttt{v2}$

Unary operation on constants
(v2 is the result of unary operation)

$$\frac{(\texttt{e1},\ \sigma) \rightarrow_b \texttt{e1'}}{(\texttt{uop e1},\ \sigma) \rightarrow_b \texttt{uop e1'}}$$

Unary operation on subexpressions

$(\texttt{v1 bop v2},\ \sigma) \rightarrow_b \texttt{v3}$

Binary operation on constants
(v3 is the result of binary operation)

$$\frac{(\texttt{e1},\ \sigma) \rightarrow_b \texttt{e1'}}{(\texttt{e1 bop e2},\ \sigma) \rightarrow_b \texttt{e1' bop e2}}$$

Binary operation on subexpressions

$$\frac{(\texttt{e2},\ \sigma) \rightarrow_b \texttt{e2'}}{(\texttt{v1 bop e2},\ \sigma) \rightarrow_b \texttt{v1 bop e2'}}$$

Binary operation on subexpressions

Operational Semantics *(small step)*

For statements: $\rightarrow \in \mathcal{C} \rightarrow \mathcal{C}$

e1, e2 – expressions
v1, v2 – values

$(\texttt{x = v}, \sigma) \rightarrow (\texttt{skip}, \sigma[x \leftarrow v])$ *Assigning a constant*

$(\texttt{x = e}, \sigma) \rightarrow (\texttt{x} = \texttt{e}', \sigma)$ *Assigning an expression*
where $(\texttt{e}, \sigma) \rightarrow_b \texttt{e'}$

$(\texttt{skip; s1}, \sigma) \rightarrow (\texttt{s1}, \sigma)$ *Sequence rule (1)*

$(\texttt{s1; s2}, \sigma) \rightarrow (\texttt{s1'; s2}, \sigma')$ *Sequence rule (2)*
where $(\texttt{s1}, \sigma) \rightarrow (\texttt{s1'}, \sigma')$

Operational Semantics *(small step)*

For statements: $\rightarrow \; \in \mathcal{C} \rightarrow \mathcal{C}$

e1, e2 – expressions
v1, v2 – values

$(\texttt{if true then s1 else s2}, \sigma) \rightarrow (\texttt{s1}, \sigma)$ ***Conditional (then)***

$(\texttt{if false then s1 else s2}, \sigma) \rightarrow (\texttt{s2}, \sigma)$ ***Conditional (else)***

$$\frac{(\texttt{e1}, \sigma) \rightarrow_b \texttt{e1'}}{(\texttt{if e then s1 else s2}, \sigma) \rightarrow (\texttt{if e' then s1 else s2}, \sigma)}$$

Conditional (expr)

$(\texttt{while e do s}, \sigma) \rightarrow$ ***While loop***
$(\texttt{if e then \{s1; while e do s\} else skip}, \sigma)$

Simple Probabilistic Language

$$
\begin{array}{lcl}
r & \in & \mathbb{R} \\
x & \in & \text{Vars} \\
\mathcal{T} & ::= & \texttt{bool} \\
\textbf{uop} & ::= & \texttt{not} \\
\textbf{bop} & ::= & \texttt{and} \mid \texttt{or} \\
\\
\mathcal{D} & ::= & \mid \mathcal{T}\, x_1, x_2, \ldots, x_n \\
\\
\mathcal{E} & ::= & \\
 & & x \\
 & & \mid c \\
 & & \mid \mathcal{E}_1 \ \textbf{bop}\ \mathcal{E}_2 \\
 & & \mid \textbf{uop}\ \mathcal{E}_1
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{S} & ::= & \\
 & & x := \mathcal{E} \\
 & & \mid x := \texttt{Bernoulli}(r) \\
 & & \mid \texttt{observe}\ (\mathcal{E}) \\
 & & \mid \texttt{skip} \\
\\
 & & \mid \mathcal{S}_1 ; \mathcal{S}_2 \\
 & & \mid \texttt{if}\ \mathcal{E}\ \texttt{then}\ \mathcal{S}_1\ \texttt{else}\ \mathcal{S}_2 \\
 & & \mid \texttt{while}\ \mathcal{E}\ \texttt{do}\ \mathcal{S}_1 \\
\\
\mathcal{P} & ::= & \mathcal{D}\ \mathcal{S}
\end{array}
$$

Claret et al. Bayesian inference using data flow analysis (FSE'13)

Probabilistic State

- **Deterministic**
- **State:** $\sigma_0 \in \Sigma_0 ::= (x_1 \to v_1, \dots x_n \to v_n)$
- x_i −variables v_i − values

- **Probabilistic**
- **State:** $\sigma \in \Sigma = \Sigma_0$
- Expressions: $A \in \mathcal{E} \times \Sigma \to \mathrm{Bool}$
- Statements: $T \subseteq (S \times \Sigma) \times (\Sigma \times [0,1])$
- **(we use notation** $(\cdot,\cdot) \xrightarrow{\cdot} \cdot$**)**

Probabilistic Assignment

- Statements: $T \subseteq (S \times \Sigma) \times (\Sigma \times [0,1])$

$(\texttt{x = v}, \sigma) \rightarrow (\texttt{skip}, \sigma[x \leftarrow v])$ ***Assigning a constant***

$(\texttt{x = e}, \sigma) \rightarrow (\texttt{x = e}', \sigma)$ ***Assigning an expression***

where $(\texttt{e}, \sigma) \rightarrow_b \texttt{e}'$

- $(x = \texttt{Bern}(p_B), \sigma) \xrightarrow{p_B} (\texttt{skip}, \sigma[x \leftarrow \text{True}])$
- $(x = \texttt{Bern}(p_B), \sigma) \xrightarrow{1-p_B} (\texttt{skip}, \sigma[x \leftarrow \text{False}])$

Probabilistic Control Flow

$$(\texttt{skip; s1},\ \sigma) \xrightarrow{1} (\texttt{s1}, \sigma)$$

Sequence rule (1)

$$\frac{(\texttt{s1},\ \sigma) \xrightarrow{p_1} (\texttt{s1'},\ \sigma')}{(\texttt{s1; s2},\ \sigma) \xrightarrow{p_1} (\texttt{s1'; s2},\ \sigma')}$$

Sequence rule (2)

Recall:

$$(x = \texttt{Bern}(p_B), \sigma) \xrightarrow{p_B} (\texttt{skip}, \sigma[x \leftarrow \text{True}])$$

$$(x = \texttt{Bern}(p_B), \sigma) \xrightarrow{1-p_B} (\texttt{skip}, \sigma[x \leftarrow \text{False}])$$

Example: Bernoulli Program

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | -- |

, 0.7), (| X | Y |
|---|---|
| False | -- |

, 0.3)

```
Y := not X;
```

(| X | Y |
|---|---|
| True | False |

, 0.7), (| X | Y |
|---|---|
| False | True |

, 0.3)

Example: Bernoulli Program

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | -- |

, 0.7), (| X | Y |
|---|---|
| False | -- |

, 0.3)

```
Y := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.21), (| X | Y |
|---|---|
| False | False |

, 0.09)

Bring together: Probability of a Trace

- ***(Finite) trace*** of executing a statement/program ($c \in \mathcal{C}$):
- $\theta = c_1 \xrightarrow{p_1} c_2 \xrightarrow{p_2} c_3 \xrightarrow{p_3} \ldots \xrightarrow{p_n} c_{n+1}$

- $c_1 = (S, \sigma_{init})$ is an initial configuration
- $c_{n+1} = (\texttt{skip}, \sigma_{final})$ is the final configuration, assuming that S terminates
- The execution took specific transitions from c_1 to c_{n+1}

- ***Probability of the trace:*** $\Pr(\theta) = p_1 \cdot p_2 \cdot \ldots \cdot p_n$

Probability of a Trace

- ***(Finite) trace*** of executing a statement/program S ($c \in \mathcal{C}$):
- $\theta = c_1 \xrightarrow{p_1} c_2 \xrightarrow{p_2} c_3 \xrightarrow{p_3} \dots \xrightarrow{p_n} c_{n+1}$

- Probability we end up in a specific configuration:

- $\Pr(c_{start}, c_{end}) = \sum_{\theta \in \mathcal{T}(c_{start}, c_{end})} \Pr(\theta)$ where $\mathcal{T}(c_{start}, c_{end})$ is a set of all traces that start in c_{start} and finish in c_{end}
- ***Aggregate trace semantics:***
- $(S, \sigma_0) \overset{p}{\Longrightarrow} (S', \sigma')$ where $\mathrm{p} = \Pr\big((S, \sigma_0), (S', \sigma')\big)$

- ***Hint:*** always check whether the distribution on traces is discrete

Example: Bernoulli Program

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | -- |

, 0.7), (| X | Y |
|---|---|
| False | -- |

, 0.3)

```
Y := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.21), (| X | Y |
|---|---|
| False | False |

, 0.09)

Observations

- $$(\text{condition True}, \sigma) \xrightarrow{1} (\text{skip}, \sigma)$$

$$(\text{condition False}, \sigma) \xrightarrow{1} \emptyset$$

(doesn't exist: no such transition*)

**alternatively: can go to special end state*

Normalization: If the probabilities $p_1 \ldots p_k$ do not sum up to 1, rescale them so that they do:

1. Compute sum: $Z = p_1 + p_2 + \cdots + p_k$
2. Rescale: $p_1' = \frac{p_1}{Z}, p_2' = \frac{p_1}{Z}, \ldots p_k' = \frac{p_k}{Z}$

Do only at the end of the program – although still expensive

Example: Bernoulli Program

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
Y := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.21), (| X | Y |
|---|---|
| False | False |

, 0.09)

```
condition ( X == True );
return Y;
```

(| X | Y |
|---|---|
| True | True |

, 0.49/0.7), (| X | Y |
|---|---|
| True | False |

, 0.21/0.7),

Example: Bernoulli Program

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
Y := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.21), (| X | Y |
|---|---|
| False | False |

, 0.09)

```
condition ( X == Y );
return Y;
```

(| X | Y |
|---|---|
| True | True |

, 0.49/0.58), (| X | Y |
|---|---|
| False | False |

, 0.09/0.58)

Example: Bernoulli Program

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
Y := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.21), (| X | Y |
|---|---|
| False | False |

, 0.09)

```
factor ( X ? 0 : -1 );
```

$e^0 = 1$

$e^{-1} = 0.37$

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.077)(| X | Y |
|---|---|
| False | False |

, 0.033)

Example: Bernoulli Program

factor => condition

(| X | Y |
|---|---|
| -- | -- |

, 1.0)

```
X := Bernoulli(0.7);
Y := Bernoulli(0.7);
```

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0.21), (| X | Y |
|---|---|
| False | False |

, 0.09)

```
factor ( X ? 0 : -∞ );
```

$e^0 = 1$

$e^{-\infty} = 0$

(| X | Y |
|---|---|
| True | True |

, 0.49), (| X | Y |
|---|---|
| True | False |

, 0.21), (| X | Y |
|---|---|
| False | True |

, 0), (| X | Y |
|---|---|
| False | False |

, 0)

Denotational Semantics

- **Syntactic Domain:** describes the syntax of the language (grammar): elements are e.g., nodes in the abstract syntax tree (AST)

- **Semantic Domain:** mathematical entities and operations on them
- For instance, sets of numbers, sets of tuples
- For probabilistic programs expectations are easy to envision

- **Meaning Function:** Translates elements of the syntactic domain to the elements and operations in the semantic domain
- Compositionality: the meaning of the AST is composite of the meaning of its nodes

Example (FSE 2013):

Algorithm $\textsc{Post}(\rho, \mathcal{S})$

Input: An input distribution ρ over the states of the program P, and a statement $\mathcal{S}$

Output: Output distribution over the states of the program P

```
switch (S)
case x := E:
    return λσ.Σ_{σ'|σ'[x←σ'(E)]=σ} ρ(σ')
case x := Bernoulli(r):
    return λσ.(r × Σ_{σ'|σ'[x←true]=σ} ρ(σ')+
                (1 − r) × Σ_{σ'|σ'[x←false]=σ} ρ(σ'))
case observe (E):
    return λσ.ite(σ(E), ρ(σ), 0)
case skip:
    return ρ
case S1;S2:
    ρ' = Post(ρ, S1);
    return Post(ρ', S2)
```

…

Example (FSE 2013):

Algorithm $\textsc{Post}(\rho, \mathcal{S})$

Input: An input distribution ρ over the states of the program P, and a statement $\mathcal{S}$

Output: Output distribution over the states of the program P

1: **switch** $(\mathcal{S})$

...

14: **case if** $\mathcal{E}$ **then** $\mathcal{S}_1$ **else** $\mathcal{S}_2$:
15: $\quad \rho_t = \lambda\sigma.\texttt{ite}(\sigma(\mathcal{E}), \rho(\sigma), 0);$
16: $\quad \rho_f = \lambda\sigma.\texttt{ite}(\sigma(\mathcal{E}), 0, \rho(\sigma));$
17: $\quad$ **return** $\lambda\sigma.(\textsc{Post}(\rho_t, \mathcal{S}_1)(\sigma) + \textsc{Post}(\rho_f, \mathcal{S}_2)(\sigma))$
18: **case while** $\mathcal{E}$ **do** $\mathcal{S}_1$:
19: $\quad \rho_p := \bot; \rho_c := \rho$
20: $\quad$ **while** $\rho_p \neq \rho_c$ **do**
21: $\quad\quad \rho_p := \rho_c$
22: $\quad\quad \rho_c := \textsc{Post}(\rho_p, \texttt{if } \mathcal{E} \texttt{ then } \mathcal{S}_1 \texttt{ else gskip})$
23: $\quad$ **end while**
24: $\quad$ **return** $\lambda\sigma.\texttt{ite}(\sigma(\mathcal{E}), 0, \rho_c(\sigma))$
25: **end switch**